Intel x86 Assembly Language Cheat Sheet

Instruction	Effect	Examples
Copying Data		
mov *src,dest*	Copy src to dest	mov $10,%eax movw %eax,(2000)
Arithmetic		
add *src,dest*	dest = dest + src	add $10, %esi
sub *src,dest*	dest = dest – src	sub %eax,%ebx
mul *reg*	edx:eax = eax * reg	mul %esi
div *reg*	edx = edx:eax **mod** reg eax = edx:eax ÷ reg	div %edi
inc *dest*	Increment destination	inc %eax
dec *dest*	Decrement destination	dec (0x1000)
Function Calls		
call *label*	Push eip, transfer control	call format_disk
ret	Pop eip and return	ret
push *item*	Push item (constant or register) to stack	pushl $32 push %eax
pop *[reg]*	Pop item from stack; optionally store to register	pop %eax popl
Bitwise Operations		
and *src,dest*	dest = src & dest	and %ebx, %eax
or *src,dest*	dest = src | dest	orl (0x2000),%eax
xor *src,dest*	dest = src ^ dest	xor $0xffffffff,%ebx
shl *count,dest*	dest = dest << count	shl $2,%eax
shr *count,dest*	dest = dest >> count	shr $4,(%eax)
Conditionals and Jumps		
cmp *arg1,arg2*	Compare arg1 to arg2; must immediately precede any of the conditional jump instructions	cmp $0,%eax
je *label*	Jump to label if arg1 == arg2	je endloop
jne *label*	Jump to label if arg1 != arg2	jne loopstart
jg *label*	Jump to label if arg2 > arg1	jg exit
jge *label*	Jump to label if arg2 ≥ arg1	jge format_disk
jl *label*	Jump to label if arg2 < arg1	jl error
jle *label*	Jump to label if arg2 ≤ arg1	jle finish
test *reg,imm*	Bitwise compare of register and constant; must immediately precede the jz or jnz instructions	test $0xffff,%eax
jz *label*	Jump to label if bits were **not** set (“zero”)	jz looparound
jnz *label*	Jump to label if bits **were** set (“not zero”)	jnz error
jmp *label*	Unconditional relative jump	jmp exit
jmp **reg*	Unconditional absolute jump; arg is a register	jmp *%eax
ljmp *segment,offs*	Unconditional absolute far jump	ljmp $0x10,$0
Miscellaneous		
nop	No-op (opcode 0x90)	nop
hlt	Halt the CPU	hlt

Suffixes: b=byte (8 bits); w=word (16 bits); l=long (32 bits). Optional if instruction is unambiguous.
Arguments to instructions: Note that it is not possible for **both** src and dest to be memory addresses.
Constant (decimal or hex): $10 or $0xff   Fixed address: (2000) or (0x1000+53)
Register: %eax %bl   Dynamic address: (%eax) or 16(%esp)
32-bit registers: %eax, %ebx, %ecx, %edx, %esi, %edi, %esp, %ebp
16-bit registers: %ax, %bx, %cx, %dx, %si, %di, %sp, %bp
8-bit registers: %al, %ah, %bl, %bh, %cl, %ch, %dl, %dh